Technical Art Document for:

Little Big Planet

Folder Structures, Naming Conventions etc etc

version 8.00
last updated 15th August 2007

for

PLAYSTATION 3

Table of Contents

Document History

This section will point out changes between one version (of the whole doc) and the next, and point out any additions to previous versions.

Version 1.00
Version 1.00 is an attempt to capture everything we already feel we know about the technical Art aspects of Little Big World (LBW), but I'm sure it will evolve over time!

Version 2.00
Have added Character Customization folder and defined categories for customizable items. Also added small section on material naming conventions.

Version 3.00
Have changed it so that themed textures live in the standard texture library, but within a themed folder, instead of putting themed textures within themed mesh folder – makes them easier to find! So, for example, within the wood folder, is a Mexico folder, that contains wood only used in Mexico.

Version 4.00
Added a section on being consistent with objects local axis and positioning within the max file – so that replacing meshes already used in levels doesn't mess things up.

Version 5.00
Added animation section

Version 6.00
Added boned objects section

Version 7.00
Added max materials/maps section

Version 8.00
Detailed customization selection/categories

Overview

This document is meant to serve as a reference manual for anyone creating art assets for LBW. It states some rules for naming of files, and naming of objects that exist within a project file, and describes the general folder structure, it might all sound a bit Nazi, but it makes life easier further down the line as the amount of assets grow – and if other people need to use a project you have made, it will make things easier to change – there is nothing more annoying than searching for an object in a large scene, when all objects are called box01, box02, box03 etc.

It also suggest solutions for common modeling problems, and outline some standards, with the hope that we can maintain a consistency with all the assets we create, regardless of who makes them.

I have used the following colours when referring to different types of things;

Red - Pay special attention to these rules, break these, and you deserve a good slap.
Blue - When referring to a folder name
Green - When referring to a filename
Magenta - When referring to an object within a project file.

Art folder Structure

Overview

Within the main project folder (Probably C:\dev\craftworld) are two major folders that relate to the art assets;

\art

This is where we put all new art assets for the project, and every single file counts, so please please please DO NOT PUT MULTIPLE ITERATIONS OF THE SAME FILE IN HERE, the place for those is in your personal work in progress folders, this folder contains a mirror image of the files that will ship with the game, so it has to remain clean. An exception to this rule is photoshop .PSD files – they are not mirrored in the '\gamedata' folder, so it is okay to put these in here if you feel it's worth keeping them for future tweaking.

\gamedata

The contents of this folder are created by PS3 Pusher (see section '**PS3 Pusher**' for more details), which basically takes the assets from the '\art' folder (See Above), does various things to them (resizing, computing bump/normal maps etc) and puts them into the '\gamedata' folder in a form that can be read by the code, using the same folder structure dictated by the '\art' folder.
In other words, you shouldn't need to ever go in here!
Here's a list of file formats that live in gamedata;

*.mol ----> Mesh files
*.anim----> Animations.
*.gmat----> Materials.
.dds----->Textures (.png, *.tga etc)

IMPORTANT POINT!;When using external tools (such as Max), you should use resources from the '\art' folder, when using the level editor, you should get resources from the '\gamedata' folder

So, now you know that, let's delve into the art folder!

\art
- \data
- \mesh_library
- \pshaders
- \texture_library

\art\data

A scary place that programmers seem to fill full of rubbish things. Don't go here.

\art\mesh_library

This is where all meshes for the project live (surprise!), and is also the place to put any textures that are exclusive to a particular mesh.
(See section '**Mesh Library**' for more info)

\art\pshaders

Various shaders live in here, as far as I can work out, these are coder generated so far, I will find out more when I can...

\art\texture_library

As the name suggests, this is the home of our textures, and as a general rule contains any texture that isn't exclusive to a specific mesh.
(See section '**Texture Library**' for more info)

Mesh Library

This is where all meshes for the project live, and is also the place to put any textures that are exclusive to a particular mesh, for example, if you have an orange mesh, that uses a dirtmap that is only used by the orange mesh, then the texture lives with it in the same folder.

As a general rule, every object has it's own folder, normally containing the mesh file/s (Some objects elements are split into two or more mesh files), and a unique texture (normally a lo-res dirtmap or unique pattern).

Meshes are split up into the following major folders;

\art
    \mesh_library
        \common_objects
        \themed
        \non_player_characters
        \player_character
        \player_character_customizations
        \player_character_hardware
        \pod

\art\mesh_library\common_objects
This is where we put objects that don't live within a particular Theme (see **Design Doc** & **Visual Doc** for more info on Themes), most objects in here will be used in various places, and some objects in here are elements that will get merged into other meshes (A nut and bolt for example).
We aren't currently using any XREF objects, but if we do, then this will the place to put them.

\art\mesh_library\themed
As the name suggests, items in here are tied to a particular Theme (African, English,Sports etc) – this is an important structure to stick to, as it will map well to releasing new sets of levels in the Future, within this folder is a set of folders, each for a theme.

\art\mesh_library\non_player_characters
This is where all of the games meanies, and other various characters live. Most objects that live in here are likely to have animations associated with them.

\art\mesh_library\player_character
This is where our central player character and all of his animation files live, as well as all customizations that are specific to the main character (Costumes, Wigs etc etc)

\art\mesh_library\player_character_customizations
All character customization bits go in here – items of clothing, wigs, etc etc. Within this folder is a folder for each category.

\art\mesh_library\player_character_hardware
All tools/character enhancement items live in here.

\art\mesh_library\pod
All here we place all things that relate to the Pod/Front end/Hub/Cloth Earth and Workshop/Personal Space part of the game.

Animations

If an object has animation files associated with it, then the animation files should be placed in a separate folder within the objects folder, and the object file that has been rigged should live in the root – this file is referred to as the 'ADAM' file, and if any rigging adjustments need to be made, this is the file to do it on.

Texture Library

As the name suggests, this is a home for textures, and as a general rule contains any texture that isn't exclusive to a specific mesh.

The textures in here are meant to be a common set, that many objects can use – so do not be tempted to change a texture in here to suit a particular object you are working on, as it could have many knock on effects for other objects.

Another thing to remember is that texture memory must be used conservatively, so we should resist temptation to have multiple copies of a texture that only vary slightly – it makes more sense to use an existing texture, and alter it via a shader (multiply it by darker texture to reduce brightness for example).

Remember that the point of textures being in here is that they can be used by many objects, so do not put textures in here that only get used by one object – the place for them is in a folder with the respective mesh.

Most textures in the texture library work well when tiled, and where appropriate, each texture will have associated bump and specular maps.

Textures are split up into the following major folders;

\art
    \texture_library
        \coarse_weaved_fabrics
        \dirt_n_scratches
        \leather_n_rubber
        \metals
        \misc
        \paint
        \paper
        \particle_effects
        \rope_string_and_stitching
        \rough_surfaces
        \soft_stuff
        \wood
        \zippers

Within these major folders, when appropriate, there are separate folders for the various themes (Mexico, Africa etc etc)

\art\texture_library\coarse_weaved_fabrics

For all things cloth like, the textures in here are all about the 'grain' or 'weave', most of the time the textures in here will be combined with another texture that describes the colour/pattern. Most of the textures in here have a diffuse, bump, and specular map, and work well when tiled – they are often used tiled many times, combined with a pattern or dirt texture that is tiled much less. This is great for creating very hi-res looking textures.

\art\texture_library\dirt_n_scratches
General dirt and scratches, that can be tiled, and used to add variation to a surface – as a general rule, when combining these textures with another, it is good to tile them a different amount to the texture you are combining it with – this helps to reduce obvious tiling when the camera is more zoomed out.

\art\texture_library\leather_n_rubber
For all your leather, rubber and vinyl needs!

\art\texture_library\metals
To satisfy all your metallic desires. Some of the textures in here work well tiled a lot, some work better when tiled less.

\art\texture_library\misc
For anything that doesn't seem to live in the other folders, and isn't worthy of a folder of it's own.

\art\texture_library\paint
Cracked and peeling paint, brush marks etc.

\art\texture_library\paper
Hand made paper textures, cardboard etc.

\art\texture_library\particle_effects
For various particle effects that don't require an associated mesh. Might make sense to move this one if we start using more mesh based particles.

\art\texture_library\rope_string_and_stitching
As the folder name suggests, all things thread, rope and string like, including chains and various stitching variations – most of the textures in here follow some strict template rules (see section '**Texture Guidelines**' for more info) – the idea being that we can easily change between one stitch pattern and another, without having to mess around with UV mapping on a mesh.

\art\texture_library\rough_surfaces
Sandpaper, brick, gravel etc.

\art\texture_library\soft_stuff
In here lives the kind of stuff you wouldn't mind landing on from a height! Sponge, polystyrene etc.

\art\texture_library\wood
If it once was part of a tree, then this is the place for it. A collection of different wood grains.

\art\texture_library\zippers
Yep, you guess it, zips, and other things that are zip like. The textures in here follow some strict template rules (see section '**Texture Guidelines**' for more info) – the idea being that we can easily change between one zipper pattern and another, without having to mess around with UV mapping on a mesh.

Naming Conventions

Overview

Right, this is the bit that artists find a struggle, but after you've read this, there is no excuse – there is a point to all this, so please try and stick to it!

General rule #1

When naming anything, be it a file, or an object within a project - Call it something that actually makes sense! Do not use names such as test1, bodge3000 etc. If it is a temporary file, then call it temp, and remember to delete it! I tend to call temp files delete_me, so I know I can delete them.

General rule #2

We want consistency with names, to help avoid problems further down the line (some systems ignore case of characters, some don't for example), and also because it looks neater when there is a list – and assets we make could end up being visible to an end user at some point, so use the following format for all names – files and objects;

*Don't use any Capital letters in names.
*Separate words with underscores.

General rule #3

Filenames can't be longer than 31 characters, otherwise it breaks the PS3 disk file system, and the upshot of the is that it will never get burnt onto the DVD/Blu ray as the disc image creator will skip it.

Correct Examples;

brick.tga
brick_burnt.tga
a_really_long_file_name.txt
an_object
an_object_bone_1
an_object_bone_2

Wrong Examples;

Bigmessy thing.tga
Test2Messedup.max
i-aM a tErrible filename.jpg

Mesh Library Naming Conventions

Filenames

As a general rule, give each object a folder of it's own, and any files that are exclusive to the object go in there with it. Name the folder the same as the object, and name the associated files in a way that makes it obvious it belongs to this object;

Example 1;

```
\art
    \mesh_library
        \common_objects
            \padlock
                padlock.max
                padlock_dirtmap.png
```

Example 2;

```
\art
    \mesh_library
        \themed
            \sports
                \skateboard
                    skateboard.max
                    skateboard_dirtmap.png
                    skateboard_wheel.max
                    skateboard_wheel_graphic.png
                    skateboard_wheel_graphic.psd (This is used for making the png file above)
                    skateboard_wheel_dirtmap.png
```

Object names within a Project

The main object should be called the same as the filename, and other objects prefixed with object name, followed by additional info.

Example;

padlock.max contains following objects;

padlock_dummy	- Dummy object that padlock is linked to.
padlock	- The actual Padlock mesh
padlock_bone_01	- Skinning Bone
padlock_bone_02	- Skinning Bone
padlock_physics	- Physics outline

Texture Library Naming Conventions

When creating more than one texture to support the same material (ie diffuse, bump, specular) – be sure to use the same name for each texture, followed by appropriate post-fix.
As elsewhere, we want filenames to describe what they actually contain – so following names are used as post-fixes to filenames ;

_diffuse	- Main colour information
_specular	-Specular/Gloss map information (Grey scale)
_bump	-Bump maps – without this postfix, bump maps won't work in game. (Grey scale)
_dirt	-Used for adding colour information to an object – in most cases , this texture is mapped to a full unwrap (Mapping Channel 1), and is multiplied by a texture that is tiled a lot – this gives us high detail, with ability to dirty things up and add a bit of colour variation. This is also the place to put any fake ambient occlusion. These textures should be very lo-res. Most objects will have an associated dirt map.
_rim	-This is used to control colour and strength of rim lighting – remember that colour of rim light is also affected by global lighting settings in the level.
_env	-Environment/Reflection maps.
_scat	-For fake Sub Surface Scattering – this maps describes a thickness.
_si	For Self illumination maps

'Munging' textures
In the name of saving memory, if we have a diffuse texture that isn't using it's alpha channel, then we can use it to store the specular map (or any other texture that only needs one channel).

So, when 'munging' textures together like this, the filename should tell us all we need to know – use the following examples as a template for this;

mytexture_diffuse.png	- Contains diffuse texture, (possibly with an alpha channel).
mytexture_diffuse_specular.png	- Contains diffuse texture, with specular map in alpha channel
mytexture_dirt_scat.png	- Contains dirt diffuse, with scat map in alpha channel.

Material Naming Conventions

As with everything else, use names that make sense! It is also very important that every material in the game has a unique name, or they will overwrite each other in the game(check my rhyming skills!). This applies to sub materials in a multi/Sub-object material also.
As a general rule, I tend to name the material the same as the object, and add post fixes for any sub materials. For example;

for a padlock, I have a material named padlock, which has two sub materials;

main material name = padlock
mat id1 = padlock_brass
mat id2 = padlock_steel

In this example, only mat id1 and mat id2 actually get added to the games material library (as padlock isn't actually a material, it's a label for the collection of sub materials!), but for the sake of neatness, this is how it should be!

Modeling Guidelines

Overview

This section attempts to set some guidelines for when modeling assets – with the hope that we can attain a good consistency with all objects in the game. Some of the points in here should be stuck to (such as mapping channel rules), others are meant to be a guide, but of course, every object has it's own set of modeling challenges, so it is impossible to come up with some magic rules that apply to every asset!

General rule #1

Use quads whenever possible, and model as low poly as possible, using a mesh smooth modifier to create higher res mesh (Don't collapse it down with high res iterations on though!) – this will help when creating LOD models later down the line.

General rule #2

Use mapping channel 1 for a complete unwrap of the model (ie, all surface of model has unique uv's), and mapping channel 2 however you like – we currently have a limit of two mapping channels – this can change if we really need it to, but keeping it like this is good for memory usage!

General rule #3

Model things to the correct scale! - See Section below on units setup and scale.

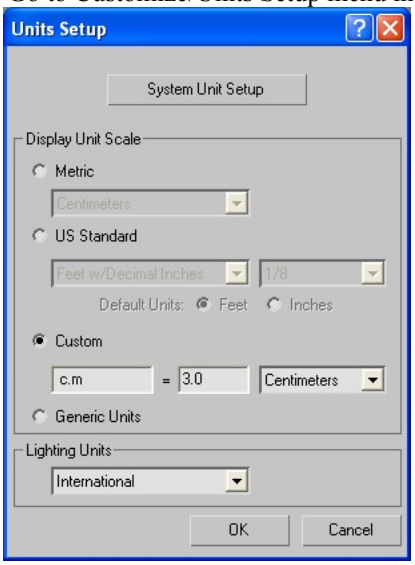
General rule #4

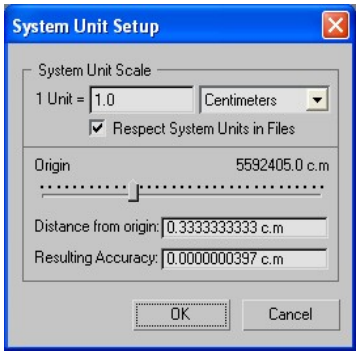
Keep the scene clean – don't have lots of old disused objects, hidden in the scene.

Units Setup and Scale

It is important that objects are modeled using the correct relative scale to each other – and we should aim to make things realistic sizes (a cup would be about 10-11 cm in real life).
Unfortunately, we managed to get confused initially with the unit set up in max, so 1 cm doesn't actually map to 1cm in our game, but follow these set up rules, and all will be fine!;

*Go to Customize/Units Setup menu in Max

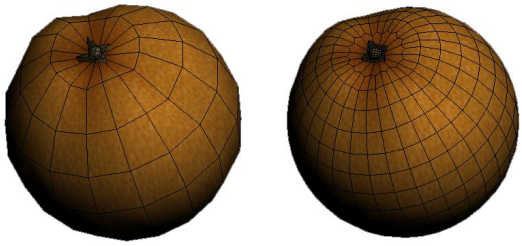




*Set up Display units, and System unit setup as follows;
*The important bits to set up are Custom 'c.m = 3.0 Centimeters', and '1 Unit = Centimeters'
*There is a 30cm/12 Inch wooden ruler (art\mesh_library\common_objects\12_inch_ruler.max) that you can look at/merge into your scene to test all is as it should be!

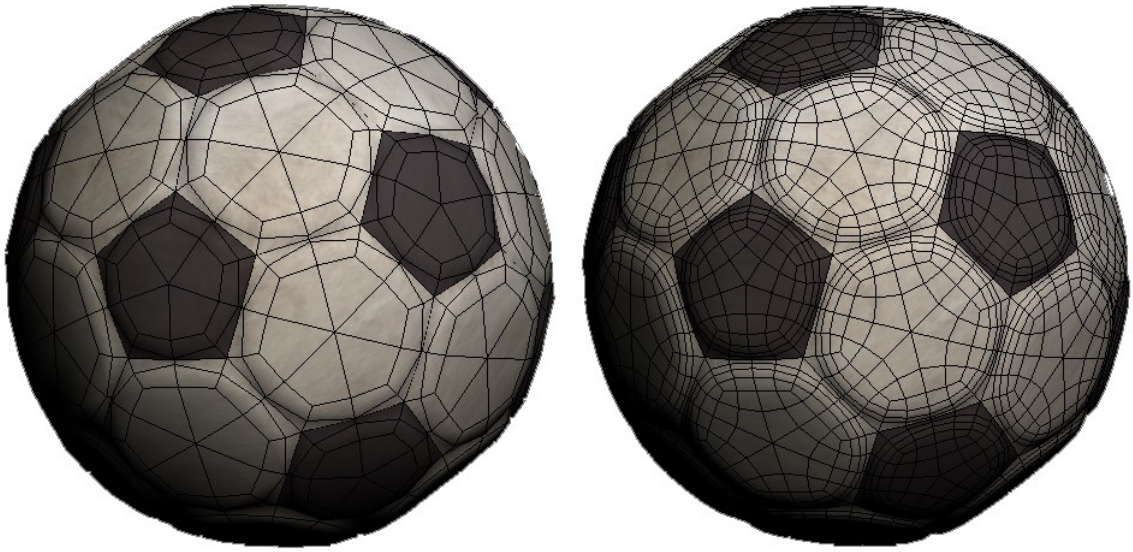
How many polygons?

We are aiming for things to look realistic as possible ,but always bear in mind not wasting polys. We will eventually want to use LOD models, so I'm approaching it like this - assume we want 3 LOD models (Lo, Medium, Hi), I model the Medium one(this is one most likely we will see most of the time), add a mesh smooth modifier to get Hi model, then do any fine detailing by moving verts/using zbrush etc, (could use this to create normal maps), and for the Lo model, can easily remove a few loops when we need to – lo models will need to be re-skinned if they have bones though.
It's almost impossible to come up with some magic rule that says how many polys an object should use , so instead, here are some examples from the game – use these as a guide;



Orange; Medium = 172 faces, High = 656 faces.

Padlock; Medium = 257 faces, High = 1028 faces



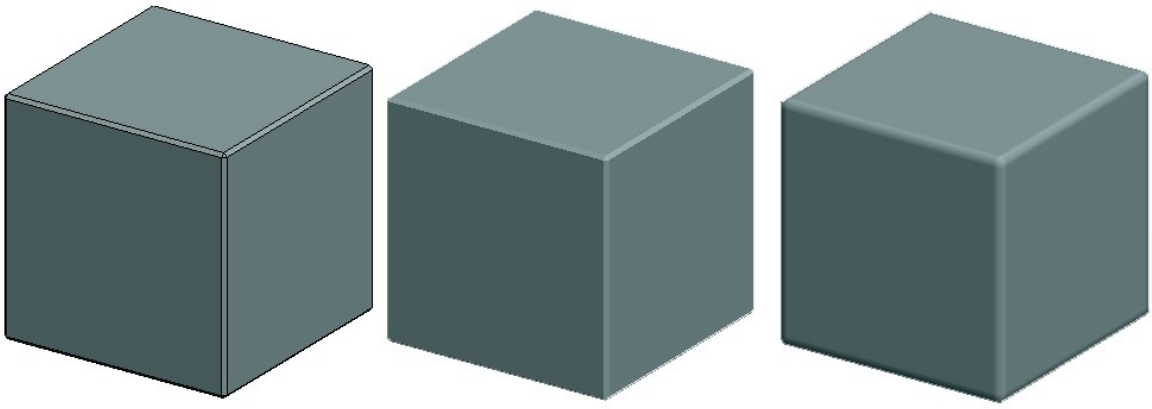
Soccer Ball; Medium = 900 faces, High = 3600 faces.

Bevels

As we are viewing real world objects closer than we normally would, we should pa special attention to things that we might ignore otherwise – Bevels for example – very few objects in real life have a molecule

crisp right angle – there is normally a more rounded edge if you look close enough – so where appropriate and noticeable – model in a bevel ;

In the above example, a different smoothing group has been assigned to beveled edge to make it crisp, if you want it to be less harsh, then the beveled edge should use smoothing group 31 – this is recognized in code, and some special things are done to cope with the smoothing (stuff that saves us having to model in an extra edge either side of bevel);



Far left cube shows geometry, middle shows different smoothing group assigned to bevel, for crisp edge, far right shows use of smoothing group 31 as it would appear in game, using same geometry.

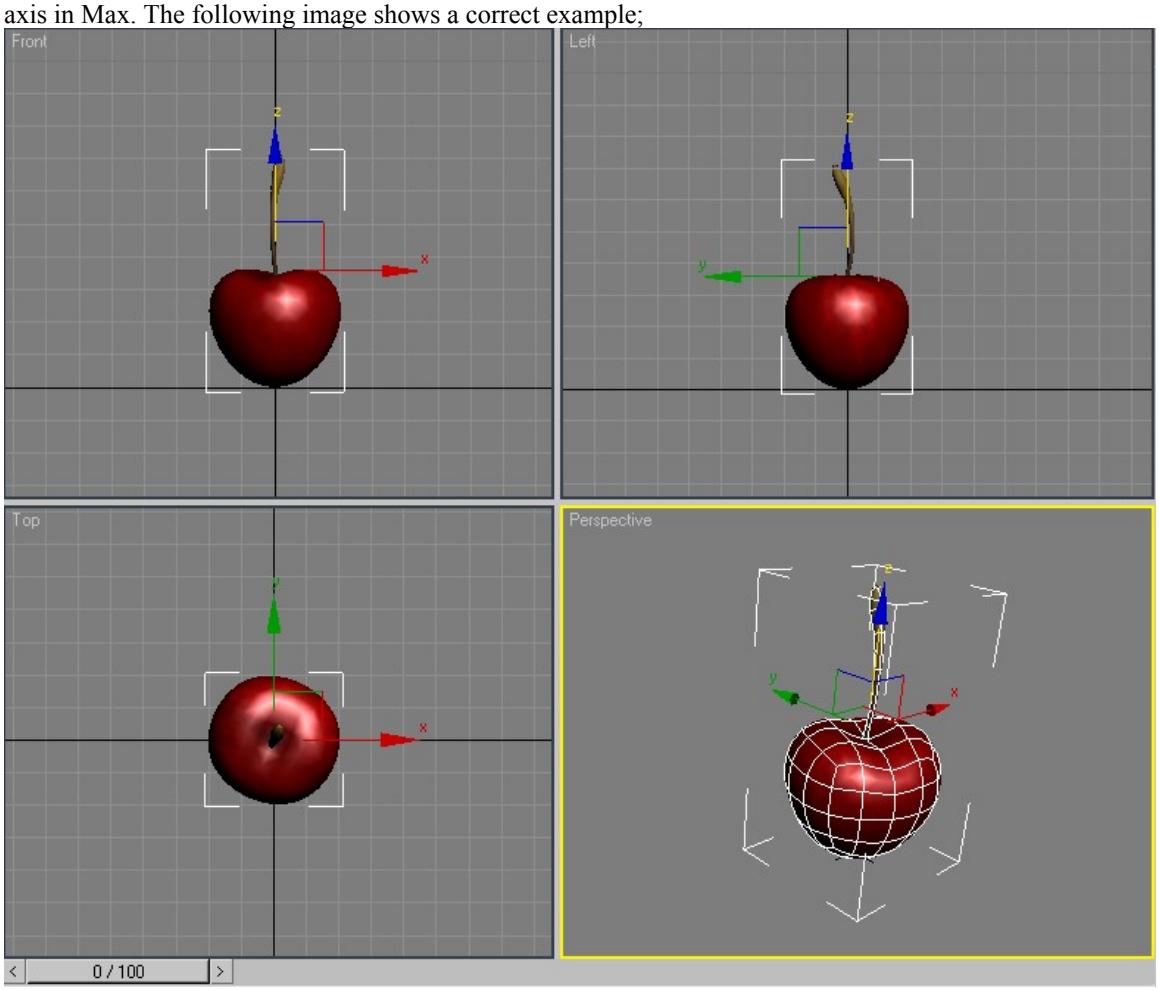
Texturing/UV Mapping

Use map channel 1 for a complete unwrap – ie all of model has unique UV co-ordinates, and map channel 2 for whatever the model requires – with the character for example, map channel 2 is used for mapping on tiled stitches and zipper.

As a general rule for texturing, to achieve hi-res looking textures, use a texture from the texture library tiled a lot, and multiply by a lo-res dirt texture to add some fake ambient occlusion and dirty bits.

Mesh Positioning within the scene and Mesh Local Axis

Sometimes we may need to re-model a mesh that has already been used in various levels – for this to work in a painless way, we need to be consistent about meshes local axis, and it's position in relation to the world axis in Max. The following image shows a correct example;



As a general rule, position a meshes local axis in the middle (as calculated by using the Pivot/Center to object function), and align the objects axis to the world (Using the Pivot/Align to World function).

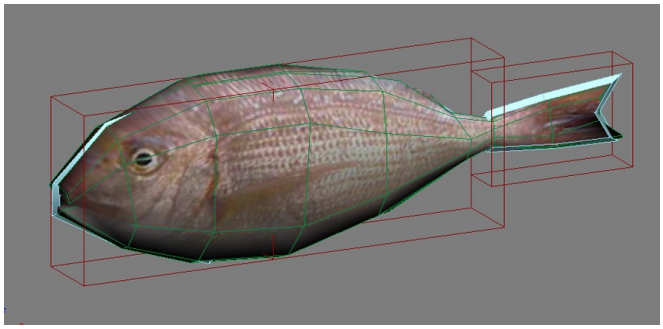
The mesh itself should 'sit' on the world axis (see above image, front view), and be centered in the Top view.

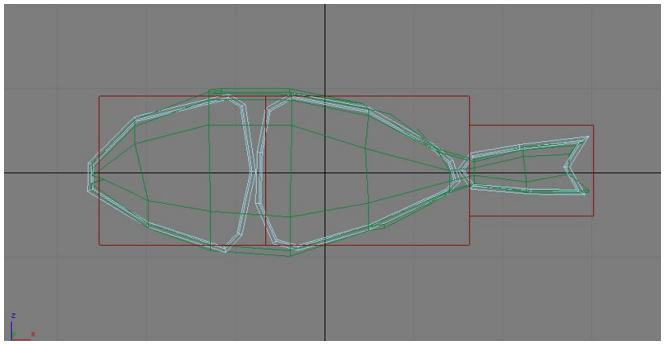
For scenes that have multiple bones etc, the root most object should follow these rules.

Sometimes, this rule might want to be broken, for example, a box, that we want at a bit of an angle (so you can see the side a bit), is nice to keep the xform intact and in line with the box (so we know how to rotate it back to being square on if need be) – in these cases, follow these rules – then rotate it as needed.

Modeling Objects with Skin and Bones

For all objects are articulated using Bones and the Skin Modifier;





1) Use boxes instead of the standard MAX Bones as shown below.
2) Place pivots manually and parent the bones into a hierarchy. Note: It is important to choose the uppermost parent bone carefully. This bone will hold both the Prendermesh and Pvalue in the PS3. Additionally, when the object is placed using Pop-it, the Tether will hold the Uppermost Parent Bone by default (eg. If it is desired that the fish is manipulated holding the tail, make the tail the Uppermost Parent Bone.)
3) Parent the physics line to the appropriate bone, making sure that they do not overlap.
4) Export Mesh using Maxecule and place joint where needed in the Editor.

Animation Guidelines

Overview

rules to remember when creating animations files

1. Animation cycles should have the same start and end poses
2. Anims that are to be blended together ie a walk into a run should share common elements
   ie. if the right foot leads and lands half way through anim, then this should be the case in both – so that when they are blended, they don't cancel each other out.

Procedural Meshes

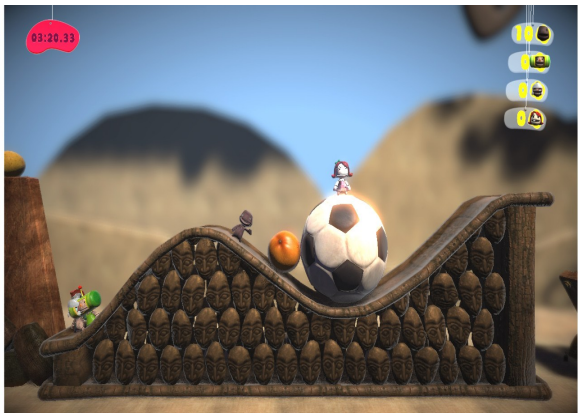
Overview

This section covers 'modular' meshes that are made to be parts of larger meshes that get put together by code – there will be some strict rules in here that if not followed, will break things in the game - So pay attention!

This hasn't been coded yet, so we don't have defined rules – but as soon as we do, this bit will get filled in, so in the meantime, here's some images to show what a procedural mesh might look like;

Some procedural bits;

These bits used procedurally in game look like this;



Character Customizations

Overview

The player can customize the character in a number of ways - the simplest way is to choose from a number of costumes, that can be mixed and matched – This section primarily deals with this customization method.

The way we are handling this is to come up with a list of categories - each separate item within a costume belongs to one of these – the player can then choose one item at a time from each category. Some items may 'use up' more than one category (a long dress for example would cover the torso, and legs categories)

Folder Structure

The folder structure in here defines the categories, they are as follows;

```
\art
    \mesh_library
        \player_character_customizations
            \face_beard          - Beards, stupid chins etc.
            \face_ears           - Silly plastic ears etc.
            \face_eyes           - Buttons, shaky eyes etc.
            \face_glasses        - Spectacles, shades, monocles etc.
            \face_mouth          - Teeth, rubber lips etc.
            \face_mustache       - Mustaches!
            \face_nose           - Silly plastic noses, duck bills, etc.
            \hair                - Wigs!_cape
            \head                - Hats, helmets, space boppers etc.
            \neck                - Scarfs, bow ties, capes etc
            \torso               - Jumpers, t-shirts, dresses etc.
            \legs                - Trousers, skirts, leg warmers etc.
            \hands               - Gloves, bracelets etc.
            \waist               - Tails, rubber ring
```

Naming Conventions

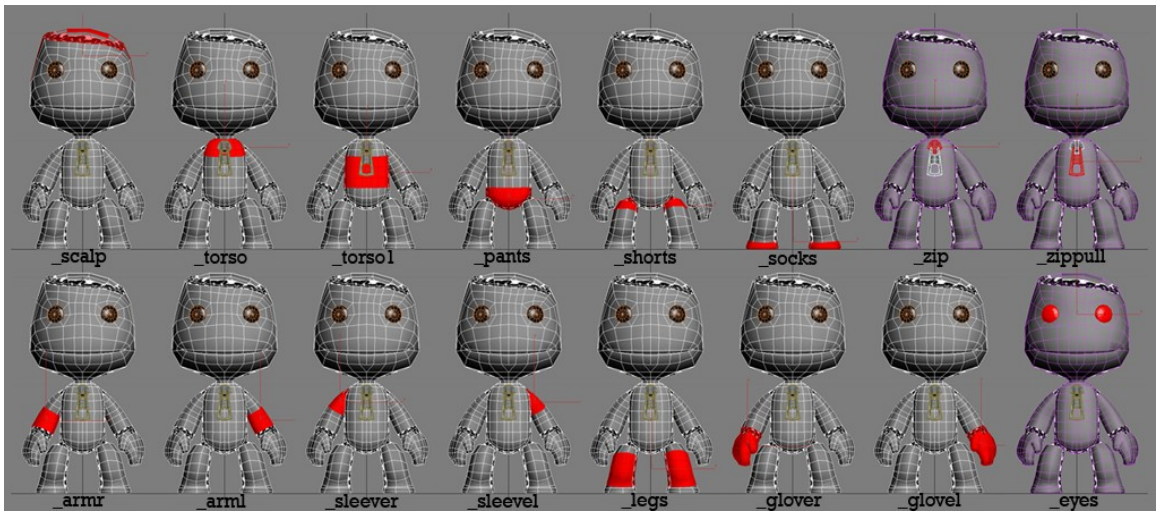
The name of the object within the max file should be the same as the filename of the project – with the following post fixes added, so the code knows how the item should behave;

```
_fb     - face_beard
_fe     - face_ears
_fi     - face_eyes
_fg     - face_glasses
_fm     - face_mouth
_ft     - face_mustache
_fn     - face_nose
_hr     - hair
_he     - head
_nk     - neck
_to     - torso
_lg     - legs
_hd     - hands
_wa     - waist
```

And finally, another post fix that refers to a named selection set of polygons in the main character file – this is for hiding polygons underneath clothing etc, to stop z-buffering problems;

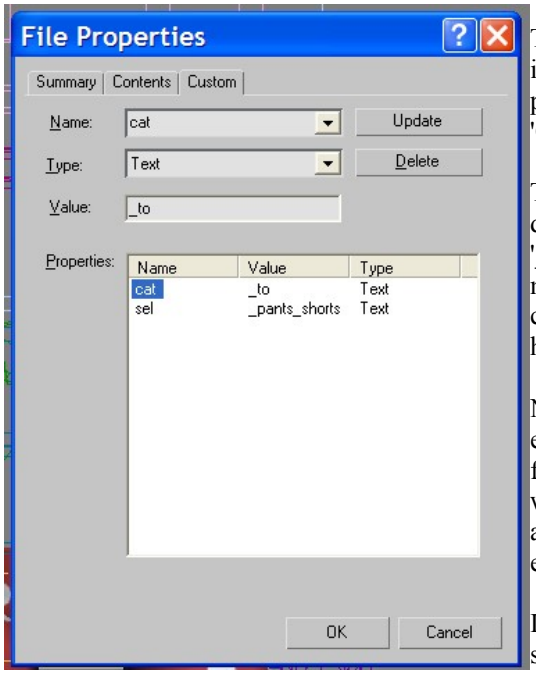
sel(name of selection set)

The names of the selection sets are detailed in the image below -



ive detailed the

Each selection has exclusive polys – ie, no polys should appear in more than one set.



The customization categories and selection set information has to be entered into the file's custom file properties – under File>File Properties, under the 'Custom' tab

Type 'cat' (category) into the 'Name' box then enter the desired categories into the 'Value' box, in this example '_to' (torso) category is used. If an item needs to cover more than one category, then simply add the extra categories into the value box ie a long dress suit might have a value like _to_lg (torso and legs)

Next, specify the selection sets the item uses, in this example the 'sel' (selections) are _pants and _shorts, as few or as many of the selections can be used – if you wanted to do a ghostly floating head customization then all the 'sel' values would be entered except the scalp and eyes!!

If a customization requires no hidden poly selection sets, such as a pair of sunglasses then no 'sel' name/value needs to be created.

Texture Guidelines

Overview

We are attempting to be as photo realistic as possible, so using photographic textures is an obvious choice – but this requires some thought in the choice and preparation of textures – consistency is something we must strive for, so here are some general rules to bear in mind;

General Rule #1
Don't use photos that have an obvious strong directional light source in them – this will make things look very messy and confusing – lighting in the game is fully dynamic – having strong lighting burnt into the texture is not compatible – so don't be tempted to paint in shadows and highlights. Look at textures already in the library, and ask yourself, does my new texture fit with these?

General Rule #2
If the texture is meant to tile a lot, then make sure it can cope with it – if the texture has an obvious large mark in it that stands out, then the chances are that it will tile very badly.

General Rule #3
Be memory conscious, and think about texture re-use – all textures in the library are there for being used by many models – textures that are exclusive to a model should only be made if really needed – when making a new texture – think – could this be a handy general texture, if so, put it in the library!

Texture Resolution

X and y dimensions of textures must be a power of 2 ie; 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 etc – they don't have to be square though.

Most textures in the library are 256x256 or 512x512, these get halved when they go through PS3 Pusher and into the game – but it's good to have higher resolution textures to hand, for promotional shots etc. Dirt maps that are exclusive to a mesh, tend to be much lower resolution – some as low as 32x32, going up to 128x128, higher for really large objects, such as a boss character.

Mixing Textures

Want to write a little bit and give some examples of using multiplied textures to get that nice hires look...

Supported Max Materials/Maps

Composite Materials
Renders the object **twice**. So expensive, but nice. Use carefully.

Composite Textures
you can also now use composite TEXTURES in any texture slot, eg diffuse, just like you use RGB Multiply. Except instead of multiplying, the alpha channel of the 2nd (and 3rd... and 4th...) textures are used to 'composite together' the texture in layers.

Mix Texture
even more powerful than Composite texture is Mix texture, which also works in any texture slot. With mix, you must specify 3 textures - a 'background' texture, a 'foreground' texture (like composite), but also a grayscale mask texture. The mask texture also has its contrast increased (like a soft threshold) with the 'use curve' button and the curve controls, which I recommend using.

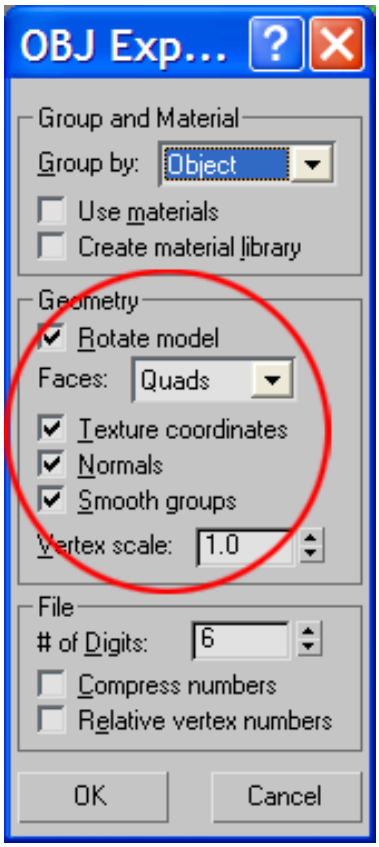
If you use a mix texture in multiple slots, it will help the shader speed if you use EXACTLY the same 'mask' texture & settings (tiling etc). I mean exactly - to the last decimal place of Uv tiling options etc.

You can of course use trees of these, eg RGB Multiplys to create more complex masks for the Mix texture. But obviously at the cost of framerate.

Normal Maps

You can now use normal maps directly out of mudbox. Here are the rules to create the correct, accurate look: (I think)

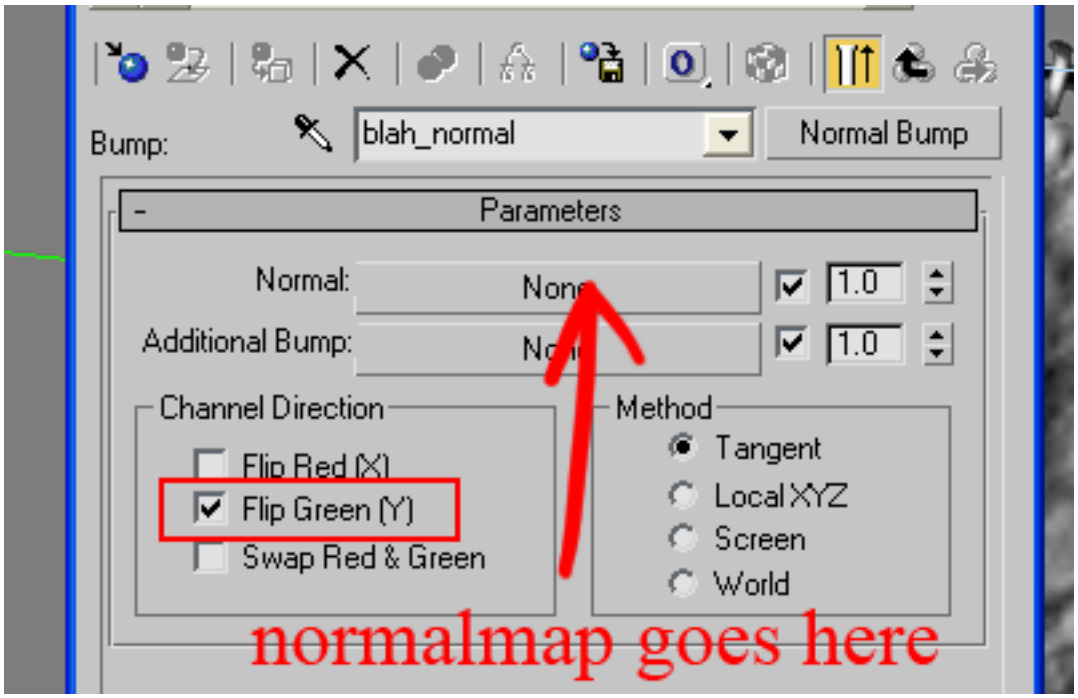
*use default tangent space options in mudbox and choose the correct options for exporting the OBJ to mudbox - see below:



* make sure the texture for the normal map has _normal in its name somewhere so ps3 pusher knows it's a normal map (same deal with _bump for bumps)

* make sure none of your other color textures use _normal in their name otherwise the new ps3pusher will munge them up :)

- create a 'Normal bump' in the bump map slot and tick the 'Flip Green (y)' box



* in MAX, set the amount of bump (next to the bump slot) to 50. Higher will exaggerate the normal map, lower will lessen it - but 50% should give it as mudbox intended...

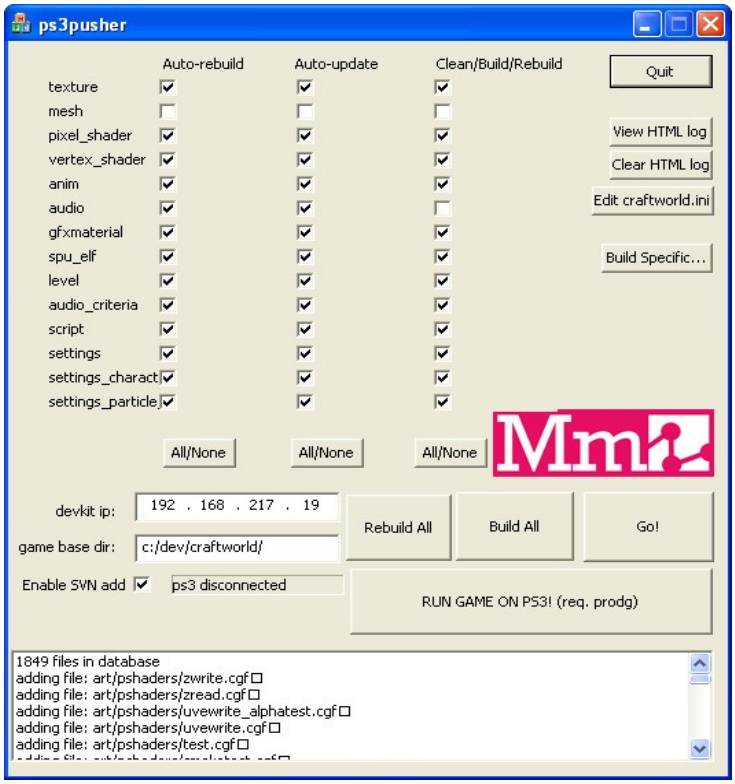
PS3 Pusher

Overview

PS3 pusher is our in house mega fantastic tool for building assets, and pushing them onto the PS3 – probably the coolest thing it does is allow real time animation/mesh/material/texture tweaking – edit any of these on P.C with the respective tool, hit save, and see how it looks in the game within a matter of SECONDS! This allows for very fine tweaking, and seeing immediate results in game, the way it should be! (note that it also 'pushes' for the P.C editor, so it's handy if not using PS3 too!)

Basics

PS3 Pusher lives in \builds\ps3pusher.exe, and looks like this;



Assuming you have a dev kit setup correctly, you can simply click on the 'RUN GAME ON PS3' button – that should get the game up and running.

The button that says 'GO' will alternate between 'GO' and 'PAUSE' when clicked on, use this if you want to stop assets from automatically updating every time you save.

You can also check individual asset types on the three columns, to specify whether they should be updated or not – The 'All/None' buttons below each column of ticks allows you to quickly tick or un-tick all of them in the column.

There is a 'bug'/feature in ps3 pusher – when you've changed your settings (the tick boxes), click the 'go/pause' button twice to restart the 'pushing' process. Otherwise sometimes it doesn't reflect the changes you made to the check boxes.

All the settings get saved when you quit ps3 pusher so set it up how you want it, then you can leave it.

Following is an explanation of each part of the interface;

'Auto-rebuild' Column
Items checked in this column will automatically be re-exported when saved or altered – this is the step that takes graphics from our \art folder, and puts them into the \gamedata folder – if you have 'mesh' checked, it will fire up max, and re-export any file you alter – so you probably want this turned off most of the time. This column basically handles any new assets going into the game.

'Auto-update' Column
Items checked in this column will get automatically sent across to PS3 whenever altered and saved – this is for assets that already exist in the game, that you are tweaking.

'Clean/Rebuild' Column
Items in this column relate directly to the 'Rebuild All' and 'Build All' buttons(see below)

'View HTML Log' Button
Clicking on this will bring up the log – a list of what's been going on with PS3 pusher, this is the place to look if assets aren't appearing in game, or are misbehaving – it the problem is obvious, it will be highlighted in red.

'Clear HTML Log' Button
Clears the log – until you clear it, it will remember all history of things going on, so it can get quite big!

'Edit Craftworld.ini' Button
Let's you edit the game .ini file, which contains various options, and setup parameters for how you want the game to run. Your IP address should be set in here, or PS3 pusher won't do the funky stuff it's good at!

'Build Specific' Button
This is to force ps3 pusher to (re)build or export a given file or files. So if you have (eg) meshes switched off for auto export, but you want to force it to-re-export one, you don't actually need to fire up max; you can just click 'build specific' and find either the max file (in the art folder), or the corresponding mol file (in the gamedata folder). Same goes for textures and all other assets.

'Rebuild All' Button
This will rebuild all the game assets that are checked in the 'Clean\Build\Rebuild' column.

'Build All' Button
This will build all assets that are checked in the 'Clean\Build\Rebuild' column, that haven't yet been built.

'Go/Pause' Toggle Button
For starting/Pausing all PS3 Pusher activity.

'Devkit IP' field
This should be set to the IP address of your Dev Kit. This should get filled in automatically . It's handy to check the 'ps3 connected' bit of text just below it; if it says 'devkit disconnected' ps3 pusher won't work until the ps3 restarts. Restarting the game on ps3 usually fixes this (basically, the ps3 talks back to ps3pusher and tells it that it about IP etc automatically).

'Game Base Dir' field
Set this to the folder that the project lives in.(Probably C:/dev/craftworld/)

'Enable SVN add' Check box
Checking this box will automatically add assets to our source control database.